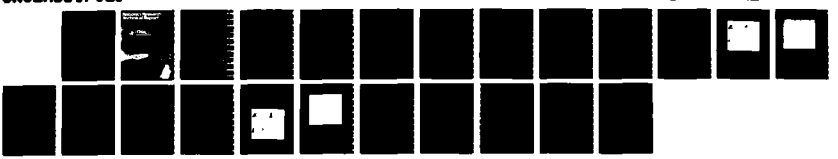
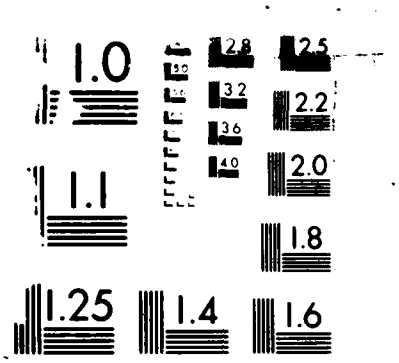
AD-A170 955 CUSTOM-MADE PYRAMIDS(U) NEW YORK UNIV NY COURANT INST OF MATHEMATICAL SCIENCES S PELEG ET AL. NOV 86 TR-255 1/1
N00014-85-K-0077

UNCLASSIFIED F/G 9/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

Robotics Research Technical Report

Custom-Made Pyramids

by

Shmuel Peleg †
Orna Federbusch †
Robert A. Hummel ‡

Technical Report No. 255
Robotics Report No. 85
November, 1986

New York University
Courant Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012

Custom-Made Pyramids

by

Shmuel Peleg †
Orna Federbusch †
Robert A. Hummel ‡

Technical Report No. 255
Robotics Report No. 85
November, 1986

†Department of Computer Science
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
peleg@hujics.bitnet or peleg@israel.csnet

‡Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, New York 10012 USA
hummel@nyu.arpa

This research has been supported by a grant from the Israel Academy of Sciences. Support for Dr. Peleg's visit to New York University was provided by ONR Grant N00014-85-K-0077.

Custom-Made Pyramids

Shmuel Peleg
Orna Federbusch
Robert Hummel

Abstract

Pyramids are data structures used to store and process images at multiple levels of resolution. The bottom level of a pyramid is used to represent data at a fine level of resolution, while higher levels of the pyramid are used for data stored at coarser levels of resolution. For example, in the Gaussian pyramid data structure, each successive level is obtained by local averaging and subsampling of the immediately lower level in the pyramid. In nearly all pyramid implementations to date, the size reduction in each dimension between levels of the pyramid is a constant factor of two.

This paper describes a scheme that permits construction of pyramids with arbitrary size reductions between levels. The reduction factors can be different in each dimension, and differ between levels, to adapt to a given application. The user can thus specify a sequence of decreasing rectangular image sizes, and construct pyramids conforming to those sizes. Further, the reduction factors can be made adaptive to region properties, enabling smooth regions to be reduced more than "busy" regions.

1. Pyramids

Pyramid data structures have proven useful in the development of image processing methods dealing with image features at varying scales of resolution. A survey on pyramid structures may be found in [1]. Typical image pyramids are formed from rectangular grid arrays whose sides have lengths that are powers of two in extent. Thus the base level might be 512 by 512, the next level up would then be 256 by 256, and each successive level reduces each dimension by a factor of two. In the Gaussian pyramid data structure, the reduction from one level to the next is accomplished by blurring the lower level (by means of convolution with a non-negative kernel) followed by a subsampling of every other pixel on every other row. In the Burt form of the Gaussian pyramid [2], each level has size of the form 2^n+1 by 2^n+1, so that in the subsampling operation left and right edge pixels are included on every second row, and that both the top and bottom rows are sampled. Nonetheless, the sampling rate is still two, and we say that the size ratio between levels is two in each dimension.

By subsampling every other pixel on every row and offsetting the samples by one pixel on successive rows, it is possible to achieve an effective sampling ratio of

$\sqrt{2}$ in each dimension [3]. In this pyramid scheme, every other level is in essence located on a 45° grid, and care must be exercised when designing kernels and other local operators on these levels. Nonetheless, the reduced reduction factor between levels results in a better chance of capturing a salient feature at an appropriate scale of resolution, at the cost of doubling the number of levels.

In this paper, we describe a method for constructing pyramids of arbitrary size at each level, so that the resolution can be reduced as needed, and not only by fixed ratios. Further, the reduction does not have to be uniform over the whole image, and can vary based on local image properties.

The basic step in the construction of the proposed pyramid scheme is a spatial resampling technique used in graphics [4,5], related to anti-aliasing. We will first describe the resampling idea in one dimension. The transformation can be applied to rectangular grids by first sampling the rows, and then sampling the columns of the result. We will then present a formulation of the sampling idea that permits the construction of pyramids using arbitrary placements of pixels (such as hexagonal grids). The central property of the sampling method is that each pixel contributes fully to the output samples, thereby minimizing sampling artifacts.

As an introduction to the resampling methods to be defined in later sections, consider the original sample points as "producers," and the new sample points that form the resampled data as the "consumers." The producers and consumers have associated positions, corresponding to the sample point locations in the domain. The amount "produced" by each producer is pre-specified, as is the consumption level of each consumer. The consumers obtain their products from a linear sum of the producers, taking contributions of fixed amounts, such that the total of all contributions by a given producer is the total amount produced at that site. The "error" of a sampling can be defined as the sum over all producer-consumer pairs of the distance between the corresponding points weighted by the contribution of the producer to the consumer. The error is minimized when the consumers use the closest suppliers to compute their values. The methods described below have certain optimality properties with respect to these measures, but we will not pursue the variational derivation any further.

2. One-Dimensional Resampling

We first consider the problem of resampling a vector $(v_0, \cdots, v_{N-1})$ of N pixels to a vector $(w_0, \cdots, w_{M-1})$ of M pixels. We treat both cases $M < N$ and $M \geq N$.

2.1. Uniform Resampling

We use the formula

$$w_i = \sum_{j=\lfloor i/\rho \rfloor}^{\lceil (i+1)/\rho \rceil - 1} r_{ij} v_j$$

where

$$r_{ij} = \begin{cases} \min\{\,1,\ \rho,\ (\,\rho\cdot(j+1) - i)\,\} & \text{if } \lfloor i/\rho \rfloor \le j < i/\rho \\ \min\{\,1,\ \rho,\ (\,(i+1) - \rho\cdot j)\,\} & \text{if } i/\rho \le j \le \lceil (i+1)/\rho \rceil \end{cases}$$

and

$$\rho = \frac{M}{N}.$$

To interpret this formula, first assume that $M \ll N$, so that ρ is small. Then nearly all r_{ij} for j in the appropriate range will have the value ρ, with the exception of the two extreme r_{ij}'s.

More generally, we can regard the output range $[0,M]$ split up into M subintervals $[i,\ i+1]$, with $i = 0,1,\cdots,M-1$. The same range is also split into N subintervals $[\rho\cdot j,\ \rho\cdot(j+1)]$, with $j = 0,1,\cdots,N-1$. The coefficient r_{ij} is the total length of the portion of the interval $[\rho\cdot j,\ \rho\cdot(j+1)]$ that intersects with the interval $[i,\ i+1]$.

For the situation with $N = 4$ and $M = 3$, to resample (v_0,v_1,v_2,v_3) with (w_0,w_1,w_2) we have the formulas

$$w_0 = \frac{3}{4}v_0 + \frac{1}{4}v_1$$

$$w_1 = \frac{1}{2}v_1 + \frac{1}{2}v_2$$

$$w_2 = \frac{1}{4}v_2 + \frac{3}{4}v_3\ .$$

as shown in Figure 1a. Figure 1b shows how the coefficients can be computed from the overlapping intervals.

Properties of this sampling method include:

- The total contribution of any given v_j to all w's is ρ; i.e., $\sum_{i=0}^{N-1} r_{ij} = \rho$.
- The sum of all contributions to any given w_i is one; i.e., $\sum_{j=0}^{M-1} r_{ij} = 1$.

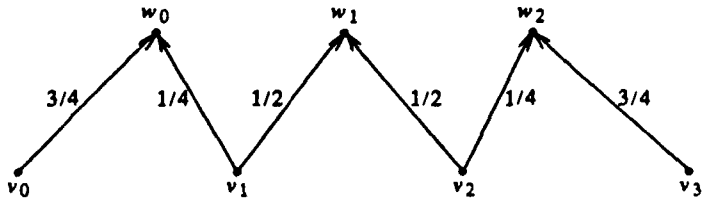


Figure 1a. Uniform resampling weights for a 4-vector to a 3-vector.

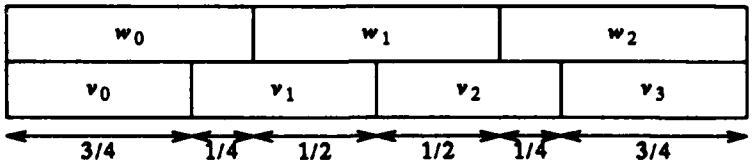


Figure 1b. Method for computing the uniform resampling weights for a 4-vector to a 3-vector, using the overlap of the intervals at different resolutions.

2.2. Weighted Resampling

In the uniform sampling given in the previous section, all input pixels make the same contribution ρ to output pixels. We now extend the resampling to permit each input pixel to have an adaptable "forward weight," so that pixel j, with value v_j, contributes a total of μ_j to the w's. These weight should satisfy the normalization condition

$$\sum_{j=0}^{N-1} \mu_j = M \ ,$$

reflecting the desire to have each of the M output pixels to receive unity in contributions from the v's. When $\mu_j = M/N$ for all j, we have the uniform sampling of the last section.

We use the same principle as in the previous section to develop a linear resampling formula

$$w_i = \sum_{j=0}^{N-1} \tilde{r}_{ij} v_j \ ,$$

but now $\tilde{r}_{ij}$ represents the length of that portion of the interval $\left[\sum_{k=0}^{j-1} \mu_k, \sum_{k=0}^{j} \mu_k\right]$ that intersects the interval $[i, i+1]$. Note that the input interval number j has length μ_j, and that the input intervals subdivide the output range $[0,M]$.

We omit the precise formulas for the $\tilde{r}_{ij}$'s, but depict the situation for $N = 4$, $M = 3$, and $\mu_0 = \mu_1 = 1$, $\mu_2 = \mu_3 = 0.5$, in Figure 2. In this case we obtain the formulas

$$w_0 = v_0$$

$$w_1 = v_1$$

$$w_2 = \frac{1}{2} v_2 + \frac{1}{2} v_3 \ .$$

We can formulate the general weighted resampling formulas by giving an interpolation formula and a sampling formula. Specifically, given a vector $(v_0, v_1, \cdots, v_{N-1})$ and the weights $(\mu_0, \cdots, \mu_{N-1})$, we form an interpolation

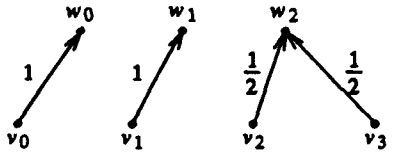


Figure 2a. Pyramid representations of the weighted sampling with weights as in the text.

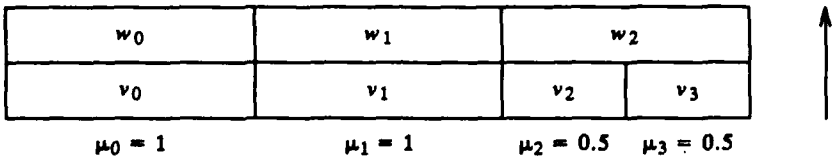


Figure 2b. Weighted resampling of a 4-vector to a 3-vector.

function

$$f(x) = \sum_{i=0}^{N-1} v_i \phi_i(x) \ ,$$

where

$$\phi_i(x) = \begin{cases} 1 \text{ if } \sum_{k=0}^{i-1} \mu_k \le x < \sum_{k=0}^{i} \mu_k \\ 0 \text{ otherwise} \ . \end{cases}$$

The samples $(w_0, \cdots, w_{M-1})$ are then obtained from the sampling formula

$$w_i = \int f(x)\psi_i(x)dx \ ,$$

where

$$\psi_i(x) = \begin{cases} 1 \text{ if } i \le x < i+1 \\ 0 \text{ otherwise} \ . \end{cases}$$

This formulation suggests a generalization, to which we return in the next section.

For the moment, we note that the $\tilde{r}_{ij}$'s satisfy

- The total contribution of any given v_j to all w's is μ_j; i.e., $\sum_{i=0}^{N-1} \tilde{r}_{ij} = \mu_j$.

- The sum of all contributions to any given w_i is one; i.e., $\sum_{j=0}^{M-1} \tilde{r}_{ij} = 1$.

3. Pyramid Construction

An input image of size M by N can be resampled to any desired output size K by L using the uniform resampling method of Section 1.1, applying first a resampling of the rows (which are N-vectors) to have lengths L, and then resampling the resulting L columns (each of which are M-vectors) to have lengths K. The result is the same if the columns are first resampled, followed by a resampling along the resulting rows. A complete pyramid structure can be built by specifying the desired sizes of all levels above the base level.

In Figure 3, we present an example of a three-level one-dimensional pyramid. Note that the values in higher levels are weighted averages of bottom-level pixels, and that the supports of the linear weighting functions can overlap in lower levels. In the immediately preceding level in a one-dimensional pyramid, with resampling as given in Section 1.1 or 1.2, support functions for adjacent pixels can overlap in at most one pixel. However, the supports from lower levels can have arbitrarily large overlaps, depending on the structure of the intermediate levels. Burt's overlapping pyramids [2] also exhibit overlapping supports, although in the one-dimensional version of his scheme (using five-tap filters), adjacent pixels in one level share supports from three pixels in the immediately preceding level (see Figure 4). Successive levels have larger overlaps in the base level. However, resampling using either the uniform resampling or weighted resampling of Section 1.1 or 1.2 yields weights determined by the sizes of the levels, and are not adjustable in the same way that the taps in the Burt pyramid can be modified.

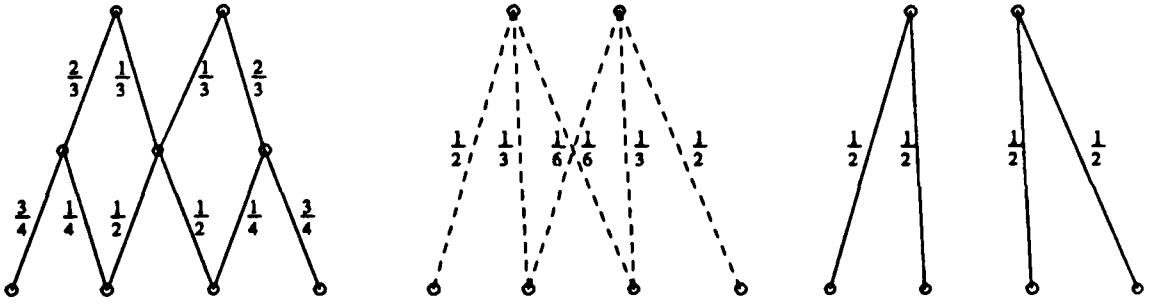


Figure 3. A comparison of a 4-3-2 one dimensional pyramid with a 4-2 pyramid structure. The leftmost pyramid shows the weights in a 4-3-2 structure, while the middle pyramid shows the effective weights from the bottom level to the top. The rightmost pyramid gives the weights for a uniform 4-2 resampling.

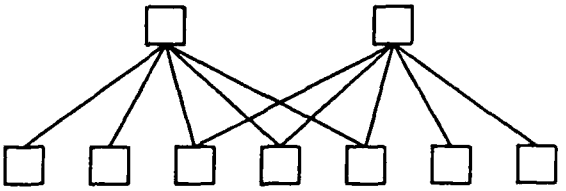


Figure 4. The Burt 5-tap pyramid.

Using the interpolation formula and sampling method formulation of resampling, however, we can generalize the uniform and weighted resampling methods to permit larger overlaps in the support functions of the resampling kernels. Rather than defining $\phi_i(x)$'s and $\psi_i(x)$'s as in Section 2.2, we instead simply assume that the interpolation kernels $\phi_i(x)$ form a partition of unity of the domain, and that the sampling kernels $\psi_i(x)$ all have unit mass. By this, we mean that

$$\sum_{i=0}^{N} \phi_i(x) = 1$$

and

$$\int \psi_i(x)dx = 1, \quad \text{for } i=0, \cdots, M.$$

In this case, resampling is still done by the formula $w_i = \sum r_{ij} v_j$, but now the r_{ij}'s are given by

$$r_{ij} = \int \psi_i(x)\phi_j(x)dx.$$

In all cases, the total of all contributions to a given pixel from all pixels at some given lower level will be unity.

The pyramid resampling scheme can be used for building successively smaller size levels (as in a Gaussian pyramid), or for expanding a level to a larger image. By combining a subsampling operation for contraction with expansion operations, the resamplings may be used for building analogs to the Laplacian pyramid data structure. The construction, analogous to Burt's formulation, proceeds as follows.

We specify a sequence of decreasing sizes (N_i by M_i) for levels i, $i = 0, 1, \cdots, \ell$ of a pyramid structure. Let G_0 be the N_0 by M_0 initial image. We construct the Laplacian pyramid as a sequence of images L_i of size N_i by M_i, $i = 0, 1, \cdots \ell$. Recursively, for $i < \ell$, G_{i+1} is obtained from G_i by resampling from an N_i by M_i image to an N_{i+1} by M_{i+1} image:

$$G_{i+1} = \mathcal{R}_{(N_i \times M_i) \to (N_{i+1} \times M_{i+1})} (G_i) ,$$

where $\mathcal{R}$ is the resampling operator. We assume, for the time being, that uniform resampling will be used. Then L_i is defined as the difference between G_i and an expansion resampling of G_{i+1}:

$$L_i = G_i - \mathcal{R}_{(N_{i+1} \times M_{i+1}) \to (N_i \times M_i)} \ (G_{i+1}) \ .$$

Finally, at the top level, we define $L_\ell = G_\ell$.

As with the Burt Laplacian pyramid, the original image G_0 can be reconstructed exactly from the L_i's using the (obvious) formulas

$$G_\ell = L_\ell$$

$$G_{i-1} = L_{i-1} + \mathcal{R}_{(N_i \times M_i) \to (N_{i-1} \times M_{i-1})} \ (G_i) \ .$$

Each level L_i represents an approximate band-pass filtered version of the original image, where the widths of the bands are determined by the choices of the relative sizes of the levels. Figure 5 shows such a custom-built Laplacian pyramid structure.

Suppose that we wish to use weighted resampling in the construction of the Laplacian pyramid. It is then desirable to invert (in some sense) the weighted resampling that was used in the contraction from level G_i to G_{i+1} for the expansion operation in the construction of L_i. The same weighted expansion should then be used in the reconstruction of G_i using the levels $L_{i+1}, \cdots, L_\ell$. A weighted expansion resampling should be used (even though it is not absolutely required for exact reconstruction) so that the levels L_i maintain their approximate bandpass characteristics. What weights should be used?

Assume that the data $(v_0, \cdots, v_{N-1})$ has been resampled using weights $(\mu_0, \cdots, \mu_{N-1})$ to obtain the M-vector $(w_0, \cdots, w_{M-1})$. Recall that the resampling is linear:

$$w_i = \sum_{j=0}^{N-1} \tilde{r}_{ij} v_j \ .$$

The matrix coefficients $\tilde{r}_{ij}$ represent the contribution fractions of coefficient v_j to coefficient w_i. We now define the "backward weights" β_i, $i = 0, \cdots, M-1$, according to

$$\beta_i = \sum_{j=0}^{N-1} \frac{\tilde{r}_{ij}}{\mu_j} \ .$$

It is easy to see that $\sum_{i=0}^{M-1} \beta_i = N$, so that the β_i's can serve as weights for resampling the M-vector $(w_0, \cdots, w_{M-1})$ to an N-vector $(v_0', \cdots, v_{N-1}')$. Each β_i represents the sum of proportions of v_j's that contributed to w_i. Accordingly, the expansion using weights $(\beta_0, \cdots, \beta_{M-1})$ leads to a vector $(v_0', \cdots v_{N-1}')$ which best approximates $(v_0, \cdots, v_{N-1})$, in the sense that there is no horizontal skewing.

For example, uniform resampling of an N-vector to an M-vector corresponds to weighted sampling where the weights are all equal to $\rho = M/N$. The backward weights are then also all equal, and have value $\rho^{-1} = N/M$. Thus uniform resampling for contraction is paired with uniform resampling for expansion.

As another example, we showed in Figure 2 the weighted sampling of a 4-vector to a 3-vector with weights (1, 1, 1/2, 1/2). The resulting backward weights are (1, 1, 2), so that the weighted backward resampling becomes

Figure 5a

Figure 5b

Figure 5. A custom-built two-dimensional Laplacian pyramid with level sizes as marked. Levels are enlarged to full size for visibility. Figure 5a show the Gaussian pyramid, while the Figure 5b contains the levels of the Laplacian pyramid. The absolute value of the data is displayed, with larger values shown more darkly.

$$v_0' = w_0 = v_0$$

$$v_1' = w_1 = v_1$$

$$v_2' = w_2 = \frac{v_2 + v_1}{2}$$

$$v_3' = w_2 = \frac{v_2 + v_3}{2}$$

as depicted by Figures 6a and 6b. The total effect of the contraction followed by the expansion is that the higher weighted nodes were exactly reconstructed, while the lower weighted nodes were blurred.

4. Adaptive Resampling

Weighted resampling, as introduced in Section 1.2, has the advantage that regions with "interesting" activity can be resampled more finely than "uninteresting" regions. This can be accomplished (in one dimension) by giving interesting pixels larger weights. In this Section, we suggest an interest operator for obtaining the resampling weights, and also discuss extensions to two dimensions and irregular tessellation grids.

4.1. One-dimensional Adaptive Pyramid

We suggest an interest operator based on the local "busyness" of the data. It has been observed that in human perception a line with higher "busyness" seems longer than a straight line segment [6], as in Figure 7. Here, we will use a smoothed absolute value of the Laplacian of the data to measure "busyness." A similar operator has been suggested for representation of intensity information by retinal receptive fields [7].

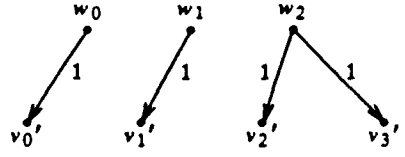


Figure 6a. Backward weights associated with the weighted resampling of Figure 2.

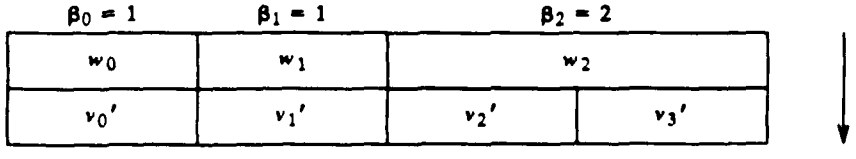


Figure 6b. Calculation of the resampling weights using the backward weights.

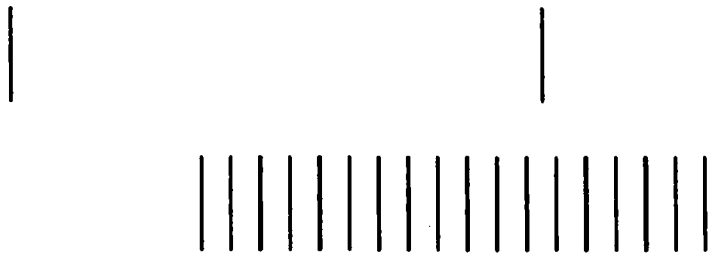


Figure 7. An illusory distortion in which the divided space appears longer than the undivided space (Oppel-Dundt illusion).

Specifically, given a signal $v_0, v_1, \cdots, v_{N-1}$, we compute the absolute Laplacian values:

$$l_i = \mid v_{i-1} - 2v_i + v_{i+1} \mid , \quad i = 1, \cdots, N-2,$$

and then smooth the results by any reasonable local averaging operator, extrapolate to include the endpoints $i = 0$ and $i = N-1$, and normalize to obtain the values μ_i, $i = 0, \cdots, N-1$. One way to accomplish the smoothing is to define an iterative weighted smoothing as follows:

$$b_{0,j} = l_j, \qquad 1 \le j \le N-2,$$

$$b_{0,0} = l_1,$$

$$b_{0,N-1} = l_{N-2} ,$$

and recursively set

$$b_{t+1,j} = \frac{1}{4}b_{t,j-1} + \frac{1}{2}b_{t,j} + \frac{1}{4}b_{t,j+1} , \qquad 1 \le j \le N-2 ,$$

$$b_{t+1,0} = \frac{3}{4}b_{t,0} + \frac{1}{4}b_{t,1} ,$$

$$b_{t+1,N-1} = \frac{1}{4}b_{t,N-2} + \frac{3}{4}b_{t,N-1} .$$

Finally, set

$$\mu_i = \frac{N \cdot b_{T,i}}{\sum_{j=0}^{N-1} b_{T,j}}$$

for some fixed integer $T > 0$ representing the amount of smoothing. The μ_i's are used as weights for resampling the v_i's.

An adaptive custom-made one-dimensional Laplacian pyramid can therefore be easily constructed. The sizes of the levels are pre-specified. The signal data is given, and weights are obtained from the busyness measures of that data.

Resampling using these weights results in the data immediately above the base level. Recomputing new busyness measures gives new weights that can be used for the next level resampling; continuing in this fashion yields a Gaussian pyramid with adaptive weights. To obtain the Laplacian pyramid, each level above the base must be expanded to the size of the previous level. This must be done using the backward weights, as described in Section 3. The resulting expanded levels are differenced with the Gaussian pyramid data at that level to obtain the Laplacian data. Reconstruction from the Laplacian pyramid structure is possible, but requires that expansion resampling using the appropriate backward weights at each level be used. This means that the complete adaptive Laplacian pyramid data structure consists of the Laplacian data at each level together with the backward weights needed for the expansion resampling at each level above the base.

4.2. Two-dimensional Adaptive Pyramids

The two-dimensional case involves technical difficulties not present in the one-dimensional case. The problems arise due to the fact that in one dimension, weights can be converted to interval lengths to decompose the domain, whereas in two dimensions it is not clear how to change the size of a rectangular region in relation to a weight and still have a complete decomposition of the domain.

One solution is to resample in each dimension separately. However, if each row or column is processed independently, then the image will lose its structure in the sense that neither connectivity or convexity of shapes will be preserved. Since the weights on one row may be independent and different from the weights on a neighboring row, nearby pixels may give principle contributions to distant pixels in the next level. Since this behavior is typically unacceptable, we suggest a scheme where all rows and all columns use the same weights. In this case, the row weights will be the average row vector of a busyness matrix, and the column weights will be the average column vector of the same matrix.

Specifically, we compute a "busyness measure" b_{ij} at each pixel (i,j) in the N by M image. Suppose that we wish to resample to an N' by M' image. Then the row weights, used to resample an M-vector to an M'-vector, are given by

$$\mu_j = \frac{M' \cdot \sum_{i=0}^{N-1} b_{ij}}{\sum_{i=0}^{N-1} \sum_{k=0}^{M-1} b_{ik}} ,$$

and the column weights, used to resample each column N-vector to an N'-vector, are given by

$$\nu_i = \frac{N' \cdot \sum_{j=0}^{M-1} b_{ij}}{\sum_{k=0}^{N-1} \sum_{j=0}^{M-1} b_{kj}} .$$

The same weights $(\mu_0, \cdots, \mu_{M-1})$ are used for resampling every row, and the weights $(\nu_0, \cdots, \nu_{N-1})$ are used for resampling the columns.

Once again, a possible method for computing "busyness values" is to smooth absolute values of the Laplacian of the image data.

As a trivial example, suppose that we wish to resample a 4 by 4 image to a 2 by 2 image, and that the busyness matrix has form

0 0 0 0
0 0 2 2
0 0 2 2
0 0 2 2

Then the row weights are given by (0, 0, 1, 1), and the column weights are (0, 2/3, 2/3, 2/3). The row resampling will then work as shown in Figure 8a, and the column resampling is as shown in Figure 8b. Note that the result is that the 3 by 2 bottom right subimage will be resampled to a 2 by 2 image, and that the other pixels will be ignored.

A contrasting example arises if the busyness matrix has the form

2 2 0 0
2 2 0 0
0 0 2 2
0 0 2 2

In this case, the row and column weights are both (1/2, 1/2, 1/2, 1/2), and the resampling for both the rows and columns will be uniform. The non-busy quadrants (the upper left and lower right portions) of the 4 by 4 image can not be contracted without distorting shapes in the remaining portions of the image.

Figure 9 shows an adaptive custom-made pyramid of an image. Note how the interesting regions in the image become stretched into larger windows at the higher levels.

5. Irregular Tessellations

The non-adaptive versions of the resampling methods given in previous sections easily extend to grid arrays other than rectangular lattices. Hexagonal grids

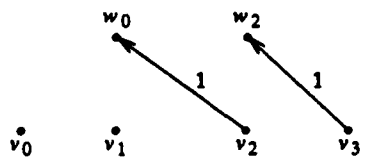


Figure 8a. Row resampling using adaptive weights of the simple example.

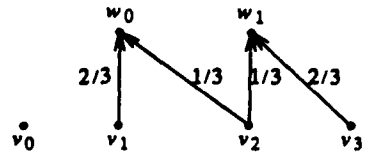


Figure 8b. Column resampling using adaptive weights of the simple example.

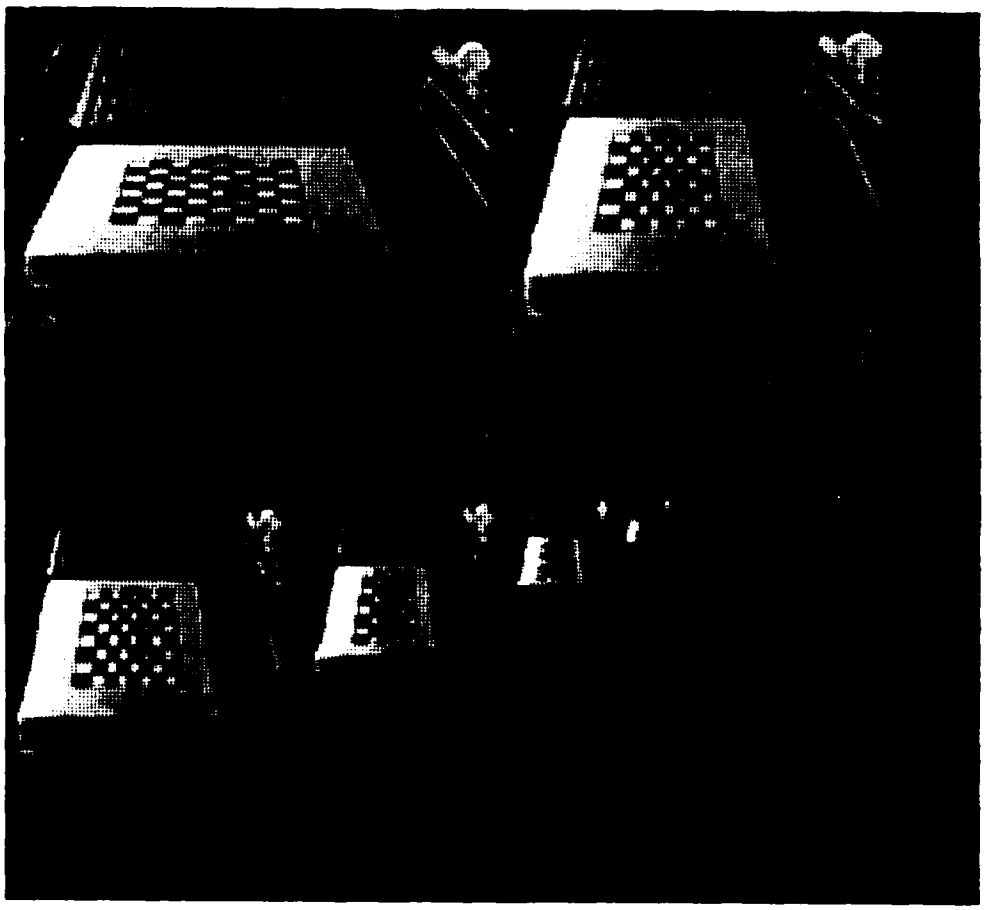


Figure 9a

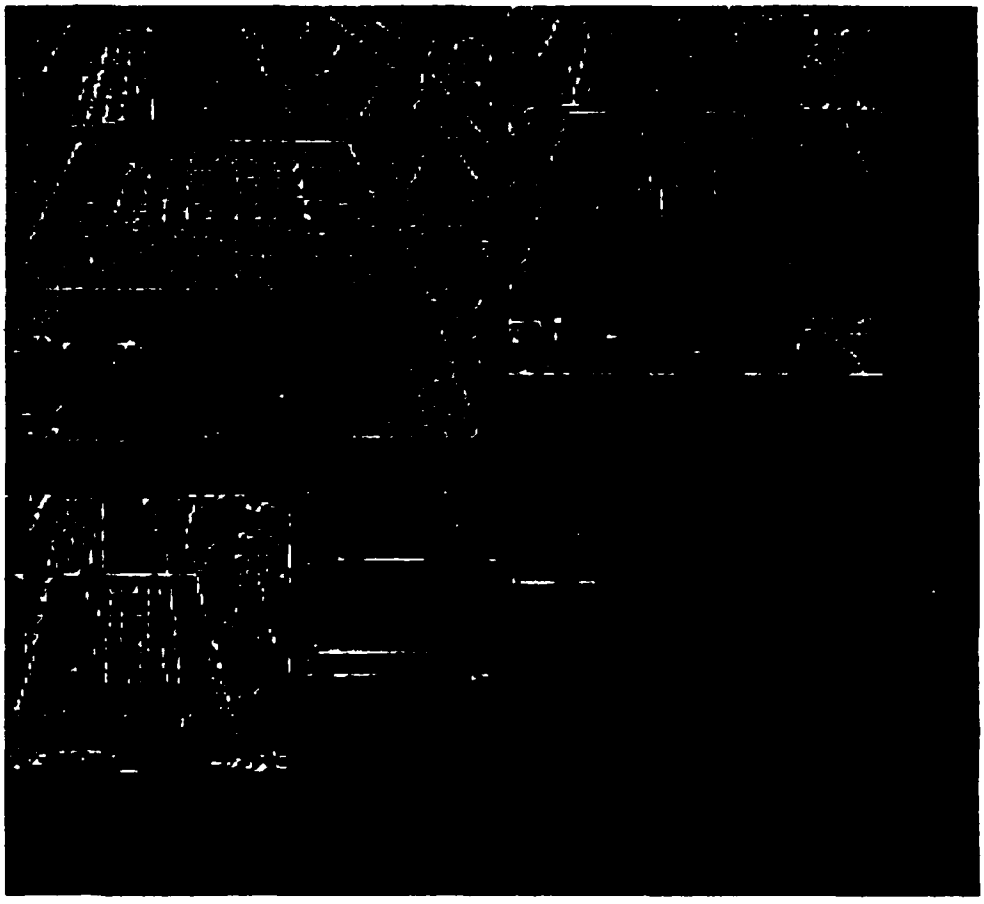


Figure 9b

Figure 9. A six level adaptive pyramid, with level sizes as marked. Figure 9a shows the levels of the Gaussian adaptive pyramid, and the Figure 9b shows the Laplacian adaptive pyramid.

are often suggested as a useful arrangement for image processing. There are hexagonal grids where the cell sizes expand with increasing distance from a central cell [8], and such grids may have a basis in human retinal cell distributions [9]. We might also imagine a random placement of sample points, with local density of sampling points prescribed by some rule (perhaps adaptive to the image data). We will first discuss the general case, and then focus on a case where sample point locations are described on a polar grid.

5.1. General case

In all cases, the locations of the sampling points are associated with either a regular tessellation or irregular tessellation (a decomposition) of the image domain into disjoint cells. Each sample v_i represents its corresponding cell C_i, and the union of all such cells covers the domain. If a tessellation is not provided as part of the sampling locations, then one can be provided by constructing the Voronoi decomposition associated with the points [10].

Given one such decomposition of the domain, say $\{C_i\}_{i=0}^{N-1}$, and another (perhaps coarser) decomposition, say $\{\mathcal{D}_i\}_{i=0}^{M-1}$, then we can define a resampling of data defined on the C-grid to the $\mathcal{D}$-grid as follows.

We are given an N-vector $(v_0, \cdots, v_{N-1})$ of data, each v_j representing a value on cell C_j. We define resampled data $(w_0, \cdots, w_{M-1})$ by

$$w_i = \sum_{j=1}^{N-1} r_{ij} v_j$$

where

$$r_{ij} = \iint \psi_i(x,y)\phi_j(x,y)dxdy \ .$$

Here we have in mind interpolation functions $\psi_j(x,y)$ that are characteristic functions of the corresponding cells C_j, and sampling functions that are characteristic functions of the cells $\mathcal{D}_i$ normalized to have unit mass. That is,

$$\psi_j(x,y) = \begin{cases} 1 & \text{if } (x,y) \in C_j \\ 0 & \text{otherwise} \ , \end{cases}$$

and

$$\phi_i(x,y) = \begin{cases} 1/\text{area}(\mathcal{D}_i) & \text{if } (x,y) \in \mathcal{D}_i \\ 0 & \text{otherwise} \ . \end{cases}$$

Area, of course, is measured by integration

$$\text{area}\ (\mathcal{D}_i) = \iint_{\mathcal{D}_i} dxdy \ .$$

More general interpolation and sampling kernels can be envisioned, but the simplest such kernels, the characteristic function of the cells as described here, should prove adequate for effective resampling.

5.2. Polar sampling

We now discuss a special sampling, the polar sampling, which is important for snapshot visual perception. In polar sampling, the sample points are located on the intersections between a set of rays and a set of concentric circles. Some models of human retinal receptor distribution describe samples points in this way, where resolution falls off with the distance (eccentricity) from the fovea. Sometimes linear fall-off of the sampling rate is assumed, which leads to placement of grid points on concentric circles whose radii increase as $\log(A+kh)$, $k=1,2, \cdots$, where A and h are constants [9]. Placement of the circles with radii increasing exponentially [11] and also simply increasing linearly [12] are also known (see Figure 10).

We can consider resampling a polar grid by reducing the number of concentric circles, or we can reduce the number of rays. When the concentric circles are reduced, we simply resample the grid points along each ray independently of the other rays. Resampling a ray is straightforward using the methods of Section 2. When the number of rays is reduced, the points on each concentric circle can be resampled independently of the other circles. To do this, the notion of one-dimensional resampling of a line segment, as described in Section 2, has to be extended to resampling along a circle. However, the methods of Section 2 extend easily, using distance based on radian measure in place of one-dimensional Euclidean distance.

If both the number of concentric circles and the number of rays are to be reduced, we can then do the resampling first by reducing the number of circles, and then by reducing the number of rays. As with two-dimensional resampling, this process is unaffected if we exchange the order of resampling. Further, the optimality properties of the weights used for resampling, alluded to in Section 1, extend to polar resampling. Indeed, because the polar grid coordinate locations can be decomposed into a cross product of circles and rays, a separable kind of adaptive resampling is possible for polar grid pyramids, similar to the adaptive two-dimensional rectangular pyramids discussed in Section 4.2.

6. Summary

Using the anti-aliasing method common in computer graphics leads to an idea for one-dimensional resampling of N points into M points. Extending this idea to two dimensions easily establishes a method for building pyramids with arbitrary

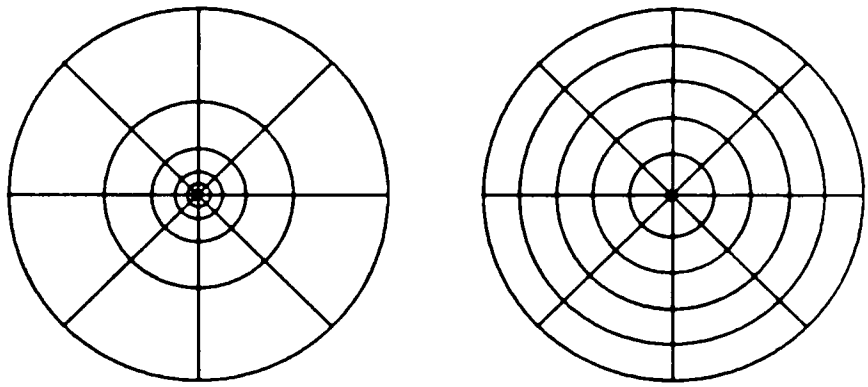


Figure 10. Polar sampling, where sampling points are on the intersection points between lines and circles. The left sampling is with exponential radius growth, while on the right the growth of the radius is linear.

sizes specified for each level. We thus see that the use of pyramids with dimensions given by powers of two is an unnecessary restriction on the construction of the pyramids. The question as to what size levels are most appropriate is left unanswered, and depends upon the application and empirical experiences.

We have also investigated the idea of adaptive resampling. The idea is easy in one dimension, and allows for arbitrary nonlinear stretching along the line to be resampled. We gave one particular busyness measure to use as a basis for deciding on the stretching. In two dimensions, things are more complicated, and we compromised by permitting only a "separable stretching," where the rows are resampled using one set of weights, yielding the same stretching for all rows, and then the columns are resampled using a single set of weights.

Finally, we note that the idea extends to irregular tessellations, where sample points can be randomly placed, or placed on polar grid or hexagonal patterns. Ideally, adaptive resampling would allow one to dynamically place additional points in regions with high busyness, but our method does not easily extend to the case of adaptive placement of resample point locations in two dimensions.

Acknowledgements

This research has been supported by a grant from the Israel Academy of Sciences. Support for Dr. Peleg's visit to New York University was provided by ONR Grant N00014-85-K-0077.

References

[1] Rosenfeld, A., *Multiresolution Image Processing and Applications*, Springer Verlag (1984).

[2] Burt, P. and T. Adelson, "The laplacian pyramid as a compact image code," *IEEE Trans. on Communications*, p. 532 (1983).

[3] Crowley, J., "A representation for visual information," CMU Robotics Institute, Ph.D. Thesis (1982).

[4] Fant, K., "A nonaliasing, real-time spatial transform technique," *IEEE Computer Graphics and Applications* **16**, pp. 71-80 (1986).

[5] Catmul, E. and A. R. Smith, "3D transformations of images in scanline order," *Computer Graphics (Proceedings SIGGRAPH 80)* **14**, pp. 279-289 (1980).

[6] Coren, S. and J. S. Girgus, *Seeing is Deceiving*, Lawrence Erlbaum Associates (1978).

[7] Zucker, Steven W. and Robert A. Hummel, "Receptive fields and the reconstruction of visual information," *Human Neurobiology*, (1985). To appear.

[8] Tanimoto, S., *Structured Computer Vision*, Academic Press, New York (1980).

[9] Schwartz, Eric, "Computational anatomy and functional architecture of striate cortex: A spatial mapping approach to perceptual coding," *Vision Research* **20**, pp. 645-669 (1980).

[10] Kirkpatrick, D. G., "Efficient computation of continuous skeletons," *Proceedings of the 20th Fundamentals of Computer Science Conference*, pp. 18-27 (1979).

[11] Anstis, S. M., "A chart demonstrating variations in acuity with retinal position," *Vision Research* **14**, pp. 589-592 (1974).

[12] Watson, A. B., "Detection and recognition of simple spatial forms," *NASA Technical Memorandum 84353*, Ames Research Center, (April, 1983).

END

5-87

DTIC